MONGO DB

Lezione I

Programma del corso

NoSQL

- Tipologie di DB NoSQL
- Differenze con RDBMS e quando usarli

MongoDB:

- Introduzione
- Join, Transaction e Pre-Join
- JSON e BSON
- Documents

Prima di iniziare…

Siamo giunti al secondo anno.

Ma quanto ricordate dall'anno scorso?

E ora si inizia

Iniziamo il corso con un introduzione e/o ripasso sul json e la sua struttura.

JSON

JSON (JavaScript Object Notation) è un semplice formato per lo scambio di dati.

Per le persone è facile da leggere e scrivere, mentre per le macchine risulta facile da generare e analizzarne la sintassi.

Si basa su un sottoinsieme del Linguaggio di Programmazione JavaScript.

JSON

JSON è un formato di testo completamente indipendente dal linguaggio di programmazione, ma utilizza convenzioni conosciute dai programmatori di linguaggi della famiglia del C, come C, C++, C#, Java, JavaScript, Perl, Python, e molti altri.

Questa caratteristica fa di JSON un linguaggio ideale per lo scambio di dati.

JSON

JSON è basato su due strutture:

Un insieme di coppie nome/valore. In diversi linguaggi, questo è realizzato come un oggetto, un record, uno struct, un dizionario, una tabella hash, un elenco di chiavi o un array associativo.

Un elenco ordinato di valori. Nella maggior parte dei linguaggi questo si realizza con un array, un vettore, un elenco o una sequenza.

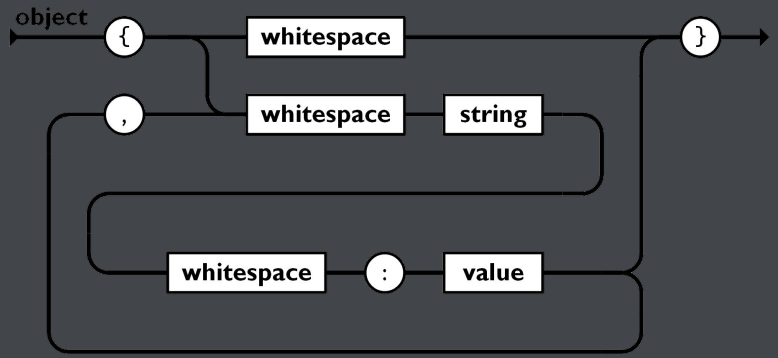
Queste sono strutture di dati universali. Virtualmente tutti i linguaggi di programmazione moderni li supportano in entrambe le forme. E' sensato che un formato di dati che è interscambiabile con linguaggi di programmazione debba essere basato su queste strutture.

JSON

In JSON, assumono queste forme:

Un *oggetto* è una serie non ordinata di nomi/valori. Un oggetto inizia con { *parentesi graffa sinistra* e finisce con } *parentesi graffa destra.*

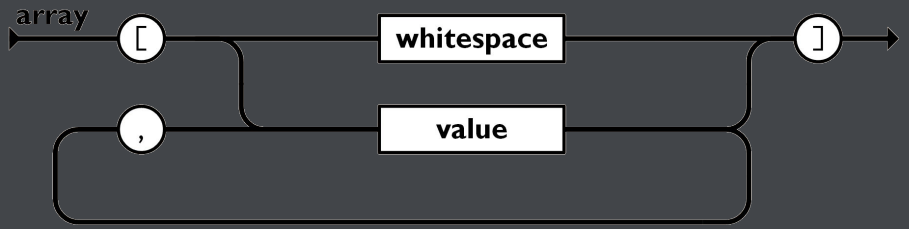
Ogni nome è seguito da :*due punti* e la coppia di nome/valore sono separata da ,*virgola*.



JSON

Un *array* è una raccolta ordinata di valori.

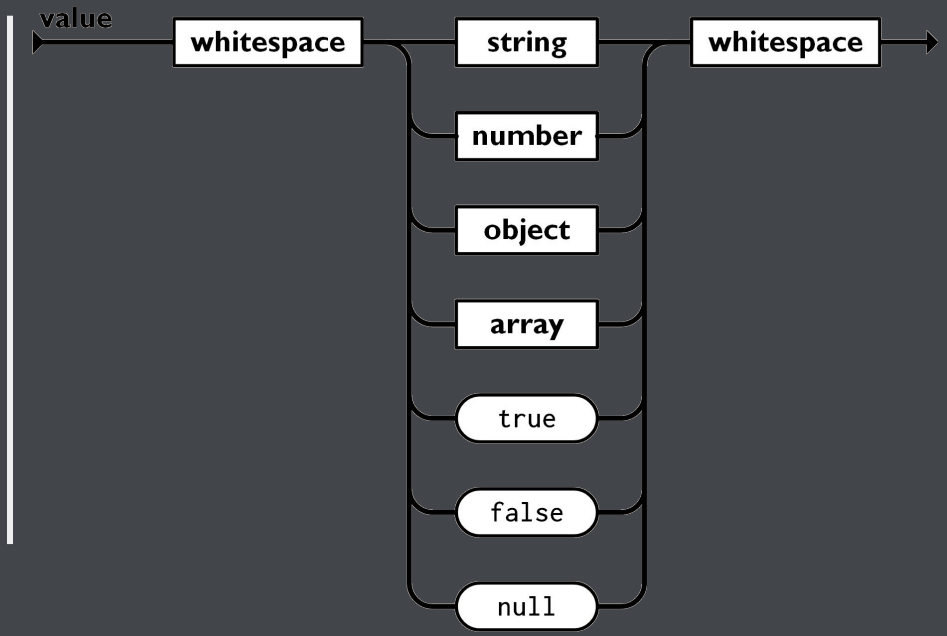
Un array comincia con [*parentesi quadra sinistra* e finisce con] *parentesi quadra destra*. I valori sono separati da , *virgola*.



JSON

Un *valore* può essere una stringa tra virgolette, o un numero, o vero *true* o falso *false* o nullo *null*, o un oggetto o un array.

Queste strutture possono essere annidate.



MONGO DB

Lezione I

Esercizio I

Esercizio I

Crea un oggetto json contenente un array denominato “studenti” con almeno 4 valori.

Questi valori devono essere altri oggetti che devono contenere dati che descrivano l’entità studente (almeno 3 in formato stringa, 2 numeri e 1 booleano.

Tempo: 10 minuti

MONGO DB

Lezione I

Esercizio II

Che cos'è NoSQL?

Definizione di NoSQL

Il termine "NoSQL" fa riferimento a tipi di database non relazionali e questi database archiviano i dati in un formato diverso dalle tabelle relazionali. Tuttavia, non è possibile eseguire query relative ai database NoSQL utilizzando linguaggi di query strutturati dichiarativi e linguaggi Query by Example, motivo per cui vengono anche definiti database "non solo SQL".

Esercizio II

Usando il json creato precedentemente aggiungi negli oggetti studenti almeno 2 sottolivelli di informazioni.

Tempo: 10 minuti

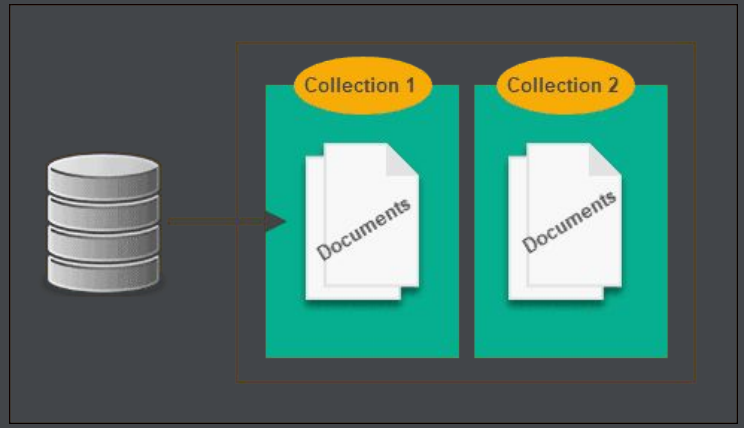
I Database NoSQL

NoSQL è un approccio al design di database che consente lo storage e l'esecuzione di query dei dati al di fuori delle strutture tradizionali che si trovano nei database relazionali.

Sebbene possa comunque archiviare i dati trovati all'interno dei sistemi di gestione di database relazionali (RDBMS), li memorizza semplicemente in modo diverso rispetto a un RDBMS.

I Database NoSQL

Invece della tipica struttura tabulare di un database relazionale, i database NoSQL ospitano i dati all'interno di una struttura dati, come ad esempio un documento JSON. Poiché questo design di database non relazionale non richiede uno schema, offre una rapida scalabilità per gestire set di dati di grandi dimensioni e tipicamente non strutturati.



I Database NoSQL

Consideriamo l'esempio della creazione di un modello di schema per il database di un semplice libro:

In un database relazionale, il record di un libro spesso viene smontato (o "normalizzato") e archiviato in tabelle distinte; le relazioni sono definite da vincoli di chiavi primarie ed esterne. In questo esempio, la tabella Libri ha colonne per ISBN, Titolo libro e Numero edizione, la tabella Autori ha colonne per IDAutore e Nome autore e infine la tabella ISBN-autore ha colonne per IDAutore e ISBN. Il modello relazionale è progettato per permettere al database di attuare l'integrità referenziale tra le tabelle del database, normalizzato per ridurre la ridondanza e generalmente ottimizzato per lo storage.

In un database NoSQL, il record di un libro è solitamente memorizzato come documento JSON. Per ogni libro, l'elemento, ISBN, Titolo libro, Numero edizione, Nome autore e IDAutore sono archiviati come attributi in un unico documento. In questo modello, i dati sono ottimizzati per lo sviluppo intuitivo e la scalabilità orizzontale.

I Database NoSQL

In che contesto si utilizzano i DB NoSQL?

Dipende!

La decisione di utilizzare un database relazionale rispetto a un database non relazionale è in gran parte legata al contesto e varia a seconda del caso di utilizzo.

MongoDB

MongoDB (da "humongous", enorme) è un DBMS non relazionale, orientato ai documenti.

Classificato come un database di tipo NoSQL, MongoDB si allontana dalla struttura tradizionale basata su tabelle dei database relazionali in favore di documenti in stile JSON con schema dinamico, rendendo l'integrazione di dati di alcuni tipi di applicazioni più facile e veloce

MongoDB

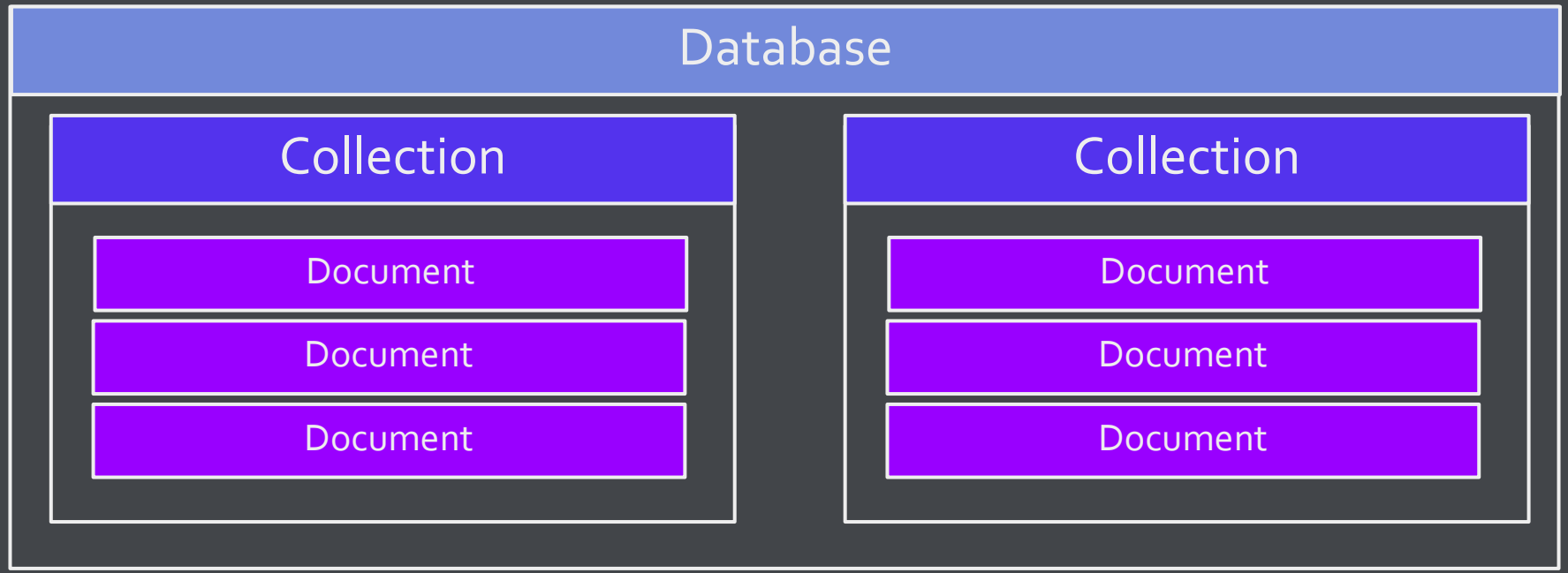
Come È Strutturato un Database MongoDB?

MongoDB è un database NoSQL senza schema. Ciò significa che non è necessario specificare una struttura per le tabelle/database come avviene per i database SQL.

Sapevate che i database NoSQL sono più veloci dei database relazionali? Ciò è dovuto a caratteristiche come l'indicizzazione, lo sharding e le pipeline di aggregazione.

MongoDB è noto anche per la velocità di esecuzione delle query. Per questo motivo è preferito da aziende come Google, Toyota e Forbes.

MongoDB


Database
Collection
Document
Document
Document
Collection
Document
Document
Document

MongoDB

Collezioni

Una collezione è un gruppo di documenti associati a un database. Sono simili alle tabelle dei database relazionali.

Le collezioni, tuttavia, sono molto più flessibili. Innanzitutto, non dipendono da uno schema. In secondo luogo, non è necessario che i documenti abbiano lo stesso tipo di dati!

MongoDB

Documenti

MongoDB ha un modello di dati documentale che memorizza i dati come documenti JSON.

I documenti corrispondono naturalmente agli oggetti del codice dell'applicazione, rendendo più semplice l'utilizzo da parte di sviluppatrici e sviluppatori.

MongoDB

Mongosh - La shell di mongo DB

mongosh è la nuova shell di MongoDB che sostituisce mongo.

mongosh è un ambiente di programmazione interattivo in cui possiamo non solo interrogare il database, ma anche eseguire qualsiasi comando Javascript.

Una volta aperta la shell possiamo vedere la lista dei db esistenti con il comando show dbs.

Creiamo il nostro primo Database.

Con il comando 'use' seguito dal nome del database andiamo a generare un nuovo db in maniera implicita.

use *nomeDB*

Da notare che il db non viene creato fino a quando non inseriamo dei dati.

MongoDB

Creare una collection

Andiamo ora a creare la prima collection. Esistono diversi modi, per il momento useremo il metodo implicito.

Per creare una collection in questo modo è necessario inserire un documento.

Il comando per inserire un documento è db.*nomeCollection*.insertOne({})

In questo modo abbiamo creato un documento vuoto. Contemporaneamente abbiamo creato sia la collection che il db.

MongoDB

Creare una collection

Una volta lanciando il comando la shell ci informa sulla riuscita dell'operazione con un messaggio:

```
{
  acknowledged: true,
  insertedId: ObjectId("65088712d3ee6ff2a7e92b72")
}
```

Questo messaggio indica che l'operazione è andata a buon fine e genera un ID unico

Per visualizzare la collection appena inserita e i documenti contenuti utilizziamo il comando

db.*nomeCollection*.find()

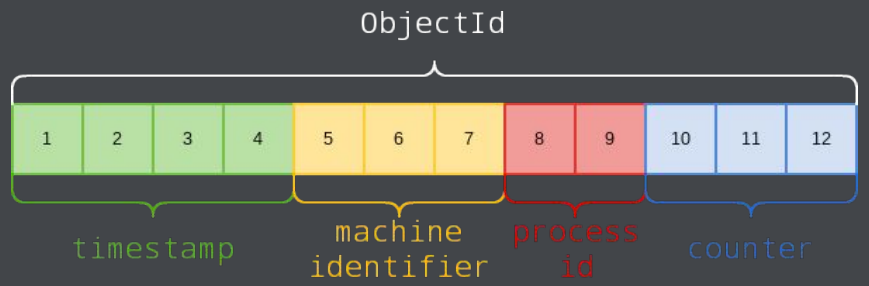
Questo comando ritorna tutti i valori contenuti dentro la collection.

MongoDB

ObjectId

Questo valore è un esadecimale composto da 12 byte:

- 4 byte rappresentano il numero di secondi dall'inizio dell'epoca Unix (1 Gennaio 1970). Per questo motivo è possibile risalire alla data di creazione di un document senza salvare un campo aggiuntivo
- 3 byte sono identificatori della macchina
- 2 byte rappresentano l'id del processo
- 3 byte sono contatori, a partire da un numero casuale



MongoDB

Creare un documento

Inseriamo ora un documento con dei valori.

db.*nomeCollection*.insertOne({ name: "Test", “description”: "Lorem ipsum" })

In questo modo abbiamo inserito un documento contenente due accoppiate chiave/valore.

MongoDB

Leggere un documento

Per leggere il contenuto di un documento il comando preposto è .find()

Se non passiamo alcun filtro restituirà tutto il contenuto della collection (esattamente come, in MySQL, una SELECT senza il WHERE).

Possiamo però indicare vari filtri.
db.*nomeCollection*.find({ name: "Test" })

In questo esempio stiamo filtrando tutti i documenti dove il *name* corrisponde a "Test".

Il comando find restituisce tutti i documenti della collection specificata che soddisfano la query indicata. La query viene indicata come parametro del metodo ed è in formato JSON.

MongoDB

Modificare un documento

È possibile in qualsiasi momento modificare uno o più document esistenti inserendo, modificando o eliminando uno o più campi. Ad esempio se volessimo aggiungere il campo *date* alla collection e aggiornare il *name*:

```
db.nomeCollection.updateOne({ name : "Test" }, { $set: {
name: "Test updated", date:
ISODate("2018-05-29T09:08:01.488Z") }})
```

Anche in questo caso la shell ci informa sulla riuscita dell'operazione e sul numero di documenti interessati:

```
{ "acknowledged" : true, "matchedCount" : 1,
"modifiedCount" : 1 }
```

MongoDB

Modificare un documento

La funzione updateOne prende in questo caso una query come primo parametro e come secondo parametro un oggetto JSON di update, che specifica i campi da modificare tramite la keyword $set.

Da notare che il metodo updateOne() lavora soltanto su un documento (seguendo l'ordine dell'id), anche se molteplici documenti soddisfano le condizioni del filtro.

MongoDB

Eliminare un documento

È possibile infine rimuovere uno o più document da una collection tramite il comando deleteOne:

```
db.nomeCollection.deleteOne({ name: "Test updated" })
```

Anche quì un messaggio ci indicherà quanti record sono stati cancellati:

```
{ "acknowledged" : true, "deletedCount" : 1 }
```

MONGO DB

Lezione I

Esercizio III

Esercizio III

Importate il json creato precedentemente dentro il vostro DB.

Scrivere tutte le istruzioni di inserimento.

Crea un'istruzione di find con almeno 2 filtri

Crea un'istruzione di delete per eliminare tutti i documenti contemporaneamente.

Ricorda: puoi e devi cercare soluzioni online!